REMARKS

The Office Action of September 2, 2004 has been received and its contents carefully considered.

Turning first to the claim objections, on page 2 of the Office Action, the present Amendment revises independent claim 11 so as to make a clear distinction between claims 1-5 and claims 11-15. Claim 11 is now directed to an article of manufacture that comprises a computer-readable medium with one or more instructions for performing a method that is specified in the claim. Since a computer-readable medium is not the same as a program debugger, regardless of what the computer-readable medium stores, it is respectfully submitted that the objection should be withdrawn.

The Office Action rejects all three of the independent claims for anticipation by an article by Wahbe et al. This reference will hereafter be called simply "Wahbe." The Office Action cites various paragraphs on pages 8 and 9 of the reference in support of the rejections. The cited paragraphs appear in a section of the reference entitled "Loop Optimization."

In general, the Office Action takes the position that Wahbe's detection of monotonic variables (page 8, column 2, paragraph 6) is equivalent to extracting the induction rate, and that Wahbe's detection of bounded writes and tagging variables with bounds (page 9, column 1, paragraph 3) is equivalent to extracting a final value for which an IV-breakpoint may be satisfied. These alleged equivalences are questionable (particularly the latter one) in view of the overall thrust of Wahbe's scheme for managing breakpoints, but it is not necessary to challenge them at the present time because a third equivalence that is alleged in the Office Action is clearly wrong.

Claim 1 recites "means for removing the IV-breakpoint, if the IV-breakpoint is satisfied in the induction variable as a present value that would be beyond the final value upon a next iteration of the loop based on the induction rate." The third equivalence alleged in the Office Action is that removing an IV-breakpoint if it is satisfied and the induction variable would be beyond the final value upon the next iteration of the loop is what Wahbe means in the fifth paragraph of column 2 on page 8. This paragraph states:

> Second, the optimizer detects write instructions that will generate a monotonic sequence of target addresses during the execution of a loop. We call such instructions *monotonic writes*. The optimizer replaces checks on monotonic writes with *range checks* in the loop pre-header. We use an efficient data structure to implement range checks. For ranges of 2^{25} bytes or less, the lookup requires at most three memory accesses. As with loop in variant checks, if a range check succeeds at runtime, the MRS will dynamically restore the eliminated write check.

It seems likely that an ordinarily skilled person would understand this paragraph to mean that Wahbe detects instructions in a loop that would cause writes to an incrementing or decrementing sequence with target addresses. Instead of checking the result of such write instructions during every iteration of the loop, he conducts what he calls "range checks," presumably meaning that whatever is stored in the sequence of target addresses is checked at the same time. If the check is positive, Wahbe restores the checks inside the loop so that execution of the loop can be examined in greater detail. An ordinarily skilled person would likely conclude from all this that Wahbe is basically saying, in the paragraph on page 8 of his article that is quoted above, is that the overall (or total) result of executing a loop which includes a write instruction that is incremented or decremented can be checked by examining what is stored in a sequence of addresses, and it is only necessary to check the effect of the instruction during each iteration of the loop if this overall check reveals a problem.

An ordinarily skilled person might well conclude that what Wahbe appears to be saying in the above-quoted paragraph on page 8 – check the overall operation of the loop and a detailed check during each iteration is not needed unless the overall check reveals a problem – is sound strategy for improving efficiency during debugging. It is not, however, the strategy employed in claim 1, where the IV-breakpoint is removed "if the IV-breakpoint is satisfied and the induction variable has a present value that would be beyond the final value upon a next iteration of the loop based on the induction rate." Nor would Wahbe's scheme suggest this to an ordinarily skilled person.

The Hanson et al reference is directed to a debugger which embeds a "nub" of itself in a program that is to be debugged. However, the Hanson et al reference neither discloses nor suggests a "means for removing the IV-breakpoint ..." in accordance with claim 1.

Independent claim 6 is a method claim which includes the following step: "if the IV-breakpoint is satisfied and the induction variable has a present value that would be beyond the final value upon a next iteration of the loop based on the induction rate, removing the IV-breakpoint." The article of manufacture defined by claim 11 comprises a computer-readable medium having one or more instructions that are executable to perform a method that includes this same step. For reasons along the lines discussed above with respect to claim 1, it is respectfully submitted that this is neither disclosed nor suggested by the references.

Since the remaining claims depend from the independent claims discussed above and recite additional limitations to further define the invention, they are patentable along with their independent claims and need not be further discussed.

For the foregoing reasons, it is respectfully submitted that this application is now in condition for allowance. Reconsideration of the application is therefore respectfully requested.

Respectfully submitted,

Allen Wood

Allen Wood
Registration No. 28,134
Customer No. 23995
(202) 326-0222
(202) 408-0924 (facsimile)

AW:rw